Application Security is one of the prime focuses when designing web applications, as they are prone to many security vulnerabilities. Oracle Application Express (APEX) has many built in security features like Authentication, Authorization i.e. access level of the user, session state protection etc.   
  
This article helps you understand Authentication schemes in APEX and how to use EBS login credentials in APEX. Authentication is a mechanism to securely identify a trusted user. It may be really simple (just enter username) or complex hash algorithm to validate username and password entered by the user.

Authentication scheme verifies user's identify before they can access your application. Once the user has been identified, APEX keep track of each user by setting the value of built-in substitution string **APP\_USER**. You can access APP\_USER using the following syntax:

* From PL/SQL: v('APP\_USER')
* As a bind variable from either PL/SQL or SQL: :APP\_USER

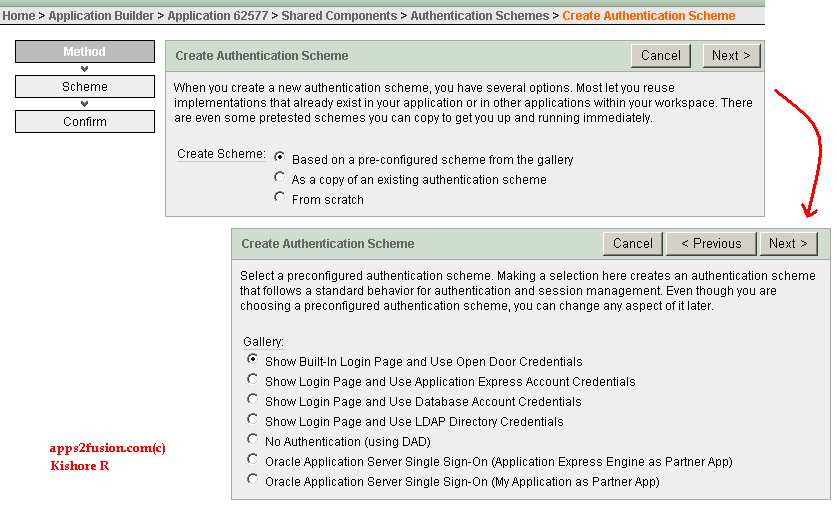
APEX Authentication Schemes are created/managed in Shared Components > Authenication Schemes (Security section) in your application.

Home>ApplicatioinBuilder>Application 62577>Shared Component

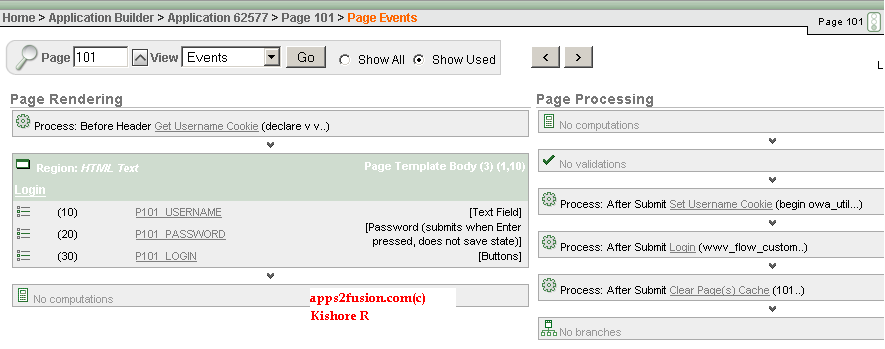
APEX comes with Preconfigured authentication schemes. When you select a preconfigured authentication scheme, APEX follows a standard behavior for authentication and session management. Preconfigured authenication schemes available are:

* **Open Door Credentials:**It enables anyone to access your application using a built-in  
  login page that captures a user name. This authentication method is useful during  
  application development.
* **Oracle Application Express Account Credentials:**It uses APEX username and password. For example, your apex.oracle.com login credentials have to be entered to run the application if you use this authentication method.
* **Database Account Credentials:**It utilizes database scheme accounts. This authentication scheme requires that a database user (schema) exist in apex database.
* **LDAP Credentials Verification:**You can configure any authentication scheme that uses a login page to use Lightweight Directory Access Protocol (LDAP) to verify the user name and password submitted on the login page.
* **DAD Credentials Verification:**This authentication scheme gets the user name from the DAD configuration or, if the account information is not stored in the DAD configuration, as the user name captured using the basic authentication challenge. This scheme also known as No Authenication.
* **Single Sign-On Server Verification:**Oracle Application Server Single Sign-On verification delegates authentication to the Oracle AS Single Sign-On (SSO) Server. To use this authentication scheme, your site must have already been registered as a partner application with the SSO server.

Below screenshot shows preconfigured authentication scheme when creating authentication scheme.



If your Oracle EBS is not integrated with Oracle SSO, you will have to create custom authentication scheme from scratch. Creating from scratch gives you complete control over your authentication interface. When defining your custom authentication scheme the following points should be noted:   
  
**1. Building a login page.**  
  
When you create a new application in APEX, a login page (page 101) is created. You can use this page as the "Invalid session page" in authentication scheme, i.e. when user session is stale or invalid, APEX redirects user to Invalid session page. You may build a custom login page instead of using default login page, and assign it to authentication scheme.  
  
Default login page (Pg. 101) is shown below. This page has user name, password text fields, a login button and few processes to complete login process.



The *Login* process takes values of entered username and password, validates them based of authentication scheme attached to APEX application, and finally redirects user to a page i.e. P\_FLOW\_PAGE parameter (default is 1) on successful validation. If you wish to redirect user to page *n,* change P\_FLOW\_PAGE parameter. Below is code for login procedure.   
  
wwv\_flow\_custom\_auth\_std.login(  
P\_UNAME => :P101\_USERNAME,  
P\_PASSWORD => :P101\_PASSWORD,  
P\_SESSION\_ID => v('APP\_SESSION'),  
P\_FLOW\_PAGE => :APP\_ID||':**1**'  
);  
  
In the login API call, you can optionally specify a *p\_preserve\_case* boolean argument. Set this to true if you don't want the username converted to upper case during credentials verification and session registration.  
  
**2. Custom Authentication Function.**  
  
This function will check the username/password and return boolean. APEX engine expects this function to have the signature (p\_username in varchar2, p\_password in varchar2) return boolean. The value of the username and password fields passed to the login API, which is called by the login page, will be passed to your function.  
  
Suppose your authentication function is *custom\_ebs\_auth,*you enter *return custom\_ebs\_auth*in authentication function field during creation. Below Code for custom\_ebs\_auth uses *fnd\_web\_sec* package to validate against Oracle EBS users.   
  
CREATE OR REPLACE function custom\_ebs\_auth (p\_username IN VARCHAR2, p\_password IN VARCHAR2)  
return boolean  
as  
begin  
if **fnd\_web\_sec.validate\_login**(p\_username, p\_password) = 'Y' then  
return true;  
else  
return false;  
end if;  
end;  
  
**3. Logout URL**This URL is used to redirect the user when logout button is clicked. Use the below URL.  
*wwv\_flow\_custom\_auth\_std.logout?p\_this\_flow=&APP\_ID.&p\_next\_flow\_page\_sess=&APP\_ID.:101:&SESSION.:LOGOUT*  
  
  
**Steps for Creating an Authentication Scheme from Scratch**  
  
As I cannot integrate my apex.oracle.com account with local Oracle EBS server, I will mimic login functionality of Sample Application (App. 100) which uses custom authentication by calling *custom\_auth*function*.*This function checks username/password in*DEMO\_USERS*table. Password column in DEMO\_USERS is encrypted using DBMS\_OBFUSCATION package and an encryption key (l\_salt variable in custom\_hash function). Remember this approach is only for demonstration of custom authentication function, not to be used for production.  
  
1. Insert users in DEMO\_USERS table.  
  
custom\_auth function encrypts entered password using key in custom\_hash function, and compares it to password column in DEMO\_USERS. So when inserting users in DEMO\_USERS, use same custom\_hash function to encrypt password. Code for inserting a2f\_admin and a2f\_user in demo\_users table. a2f\_admin user has ADMIN\_FLAG set to 'Y'.

declare  
l\_username varchar2(4000) := 'A2F\_ADMIN';  
l\_password varchar2(4000) := 'welcome1';  
l\_username1 varchar2(4000) := 'A2F\_USER';  
l\_password1 varchar2(4000) := 'welcome2';  
l\_salt varchar2(4000) := '4BS4EJ1R3L4UNRWZKPCX0HK6MTJ5YB';  
begin  
-- ADMIN User  
l\_password := custom\_hash(l\_username, l\_password);  
insert into demo\_users values (DEMO\_USERS\_SEQ.nextval, l\_username, l\_password, SYSDATE, NULL, 'Y', NULL, 'Y');  
-- Trigger BI\_DEMO\_USERS overrides admin\_flag to 'N' for new users  
-- So update is required for A2F\_ADMIN  
update demo\_users set admin\_user = 'Y'  
where user\_name = 'A2F\_ADMIN';  
-- User  
l\_password1 := custom\_hash(l\_username1, l\_password1);  
insert into demo\_users values (DEMO\_USERS\_SEQ.nextval, l\_username1, l\_password1, SYSDATE, NULL, 'Y', NULL, 'N');  
commit;  
end;  
  
2. Create custom authentication scheme.  
  
Go to Shared Components > Authentication Scheme (Security Scheme) > Create. Select From scratch in create scheme.

Click on Next.

Select Page in This Application and Page 101 Login > Click on Next.

Select Use my Custom function to Authenticate > Enter return custom\_auth >Click on Next

Click on Create Schema.

3. Make authentication scheme Current  
  
Once the authentication scheme is created, it has to be assigned to application. This is done from *Change Current* section in Authentication Scheme.

4. Run the application. I've changed login page to include some html text.

Login with your username and Password.

Please use any of the below user logins

Admin Login  : a2f\_admin/welcome1

User Login : a2f\_user/welcome2

**URL for the application:**My application can be accessed using the url <http://apex.oracle.com/pls/apex/f?p=62577:1>

**Packaged Application:**  
  
My Packaged applications are created using APEX 3.2 version, you can only import them into APEX with same version. This packaged application has supporting objects i.e. table and sample data, along with apex application. You can import and run it without going through the above steps.  
  
[Download Packaged Application](http://www.apps2fusion.com/training_demo/kishorer/apex/08_cust_auth/apex_tut03_app_with_cust_auth.zip)

The zip file has sql files for application (apex\_tut03\_app.sql) and image (apex\_tut03\_img.sql).  
  
[Video for deploying packaged application (2:41 min)](http://www.apps2fusion.com/training_demo/kishorer/apex/03_tut01/apex_tut01_step03/apex_tut01_step03.html). This video is applicable for deploying packaged applications for my next articles as well.

The next article will be on Authentication vs Authorization, how Authorization is used in APEX.